JOIN (Jurnal Online Informatika)
Volume 3 No. 1 | Juni 2018 : 29-35
DOI: 10.15575/join.v3i1.179

Implementasi Komputasi Paralel untuk Optimalisasi Komputasi pada Aplikasi Transliterasi Huruf Latin ke Aksara Jawa

A'la Syauqi[1], Anis Nurul Hidayah[2]
[1,2]Jurusan Teknik Informatika Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim
Malang, Indonesia
[1]syauqi@ti.uin-malang.ac.id, [2]anisnurul20@gmail.com

***Abstract*-Javanese script is one of the ancestral heritage of Indonesia that must be preserved. Nowadays javanese script is increasingly abandoned, since latin writing is often used in daily activities. This is become worse by student difficulties of learning to read and write javanese script because the methods are less attractive for example teacher centered learning. One of solution to solve this problem is interactive latin to javanese script transliteration application. Research on the latin to javanese script transliteration has been produced various applications such as JawaTex, Hanacaraka, and Pandawa. From these three applications is showed that the Pandawa application with decision tree method has the highest level of accuracy. However, the computational time required by this application method is also highest than other applications. The more transliterated text is conduct the longer process time is used. This research aims to shorten Pandawa application computing time with parallel computing method without modifies decesion tree algorithm. Parallel computing is implemented with multithread on java programming language to perform the process of latin into javanese script transliteration simultaneously. Experimental results obtained that the implementation of multithread in decicion tree method for latin into javanese script transliteration reduce the average computation time up to 97.05%**

***Kata kunci*- javanese script, parallel computing, transliteration, multithread**

Abstrak— Aksara jawa merupakan salah satu warisan leluhur bangsa Indonesia yang harus dilestarikan. Aksara jawa kini semakin ditinggalkan, karena tulisan latin lebih sering digunakan dalam kegiatan sehari-hari. Hal ini diperburuk oleh sulit diterimanya pembelajaran baca tulis aksara jawa dikarenakan metode pembelajaran yang kurang menarik contohnya pembelajaran yang terpusat pada guru. Salah satu solusi untuk untuk penyelesaian permasalahan ini adalah dengan aplikasi transliterasi aksara latin ke aksara jawa interaktif. Aplikasi dari hasil penelitian transliterasi huruf latin ke aksara jawa telah banyak dihasilkan, antara lain JawaTex, Hanacaraka, dan Pandawa. Dari ketiga aplikasi tersebut ditunjukkan bahwa aplikasi Pandawa dengan metode *decision tree* mempunyai tingkat keakurasian paling tinggi. Namun demikian, waktu komputasi yang dibutuhkan oleh metode dalam aplikasi ini juga paling tinggi daripada dua aplikasi lainnya. Hal ini berbanding lurus dengan banyaknya teks yang ditransliterasikan, semakin banyak teks yang ditransliterasikan maka akan semakin lama pula waktu prosesnya. Penelitian ini bertujuan untuk mempersingkat waktu komputasi aplikasi Pandawa dengan metode komputasi paralel tanpa modifikasi algoritma *decision tree*. Komputasi paralel diimplementasikan dengan *multithread* pada pemrograman java untuk melakukan proses transliterasi huruf latin ke aksara jawa secara simultan. Dari pengujian diperoleh hasil bahwa implementasi multithread pada metode *decicion tree* untuk transliterasi huruf latin ke aksara jawa ini berhasil dapat mengurangi waktu komputasi rata-rata hingga 97.05%.

***Keywords*-aksara jawa, komputasi paralel, transliterasi, multithread**

I. PENDAHULUAN

Salah satu diantara kekayaan budaya Indonesia adalah adanya bahasa daerah. Bahasa Jawa merupakan salah satu bahasa daerah dengan jumlah penutur yang paling banyak. Berdasarkan sensus tahun 2000 jumlah penutur Bahasa Jawa mencapai 84,3 juta orang. Namun data tersebut bertolak belakang dengan kenyataan bahwa pada saat ini banyak kalangan muda yang enggan menggunakan Bahasa Jawa yang disebabkan oleh beberapa hal diantaranya penggunaan bahasa gaul yang lebih populer di kalangan pemuda[1], semakin berkurangnya pemahaman kosakata Bahasa Jawa oleh pemuda[2], dan hal-hal lain yang secara tidak langsung mempengaruhi remaja lebih memilih menggunakan Bahasa Indonesia daripada Bahasa Jawa yaitu tingkat pendidikan dan jenis pekerjaan orangtua, tempat asal, jenis kelamin, dan lawan bicara[3]. Sedangkan secara umum orang tidak lagi menggunakan Bahasa Jawa salah satunya disebakan karena migrasi[4]. Selain itu penetapan Bahasa Indonesia sebagai Lingua Franca juga mempengaruhi penggunaan Bahasa Jawa di kalangan masyarakat luas[5].

Upaya Pemerintah dalam pelestarian bahasa daerah telah dinyatakan dalam Undang-Undang Dasar 1945 Pasal 32 bahwa bahasa daerah yang masih digunakan oleh masyarakat penuturnya dipelihara oleh Negara. Di

samping itu, dalam Undang-Undang Nomor 32 Tahun 2004 tentang Pemerintahan Daerah Pasal 22 huruf n dinyatakan bahwa dalam menyelenggarakan otonomi, daerah mempunyai kewajiban melestarikan nilai sosial budaya[6]. Di bidang pendidikan kebijakan untuk pelestarian kebudayaan daerah dinyatakan dalam UU Sisdiknas No 20 tahun 2003 Bab X Pasal 37 yang mewajibkan muatan lokal dalam kurikulum pendidikan dasar dan menengah. Sebagai bentuk pelaksanaannya misalnya dengan diwujudkannya Keputusan Gubernur Jawa Tengah Nomor: 423.5/5/2010 tentang Kurikulum Muatan Lokal (Bahasa Jawa) untuk Jenjang Pendidikan SD/SDLB/MI dan SMP/SMPLB/MTs Negeri dan Swasta Provinsi Jawa Tengah.

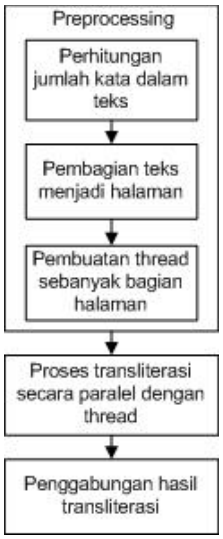
Namun demikian pembelajaran Bahasa Jawa khususnya materi baca tulis aksara jawa dinilai masih sulit diterima oleh siswa. Hal tersebut disebabkan oleh metode pembelajaran yang terpusat pada guru[7], berbasiskan buku teks dan lembar kerja siswa[8], strategi pembelajaran yang kurang menarik[9], dan kurangnya waktu yang disediakan[10]. Hal tersebut mengakibatkan kemampuan baca tulis siswa rendah[11]. Salah satu upaya dalam penyelesaian masalah pembelajaran baca tulis aksara jawa maka telah dibangun aplikasi transliterasi huruf latin ke aksara jawa interaktif yang mampu mengubah kalimat yang tertulis dengan huruf latin ke aksara jawa.

Penelitian tentang aplikasi transliterasi aksara jawa telah banyak dilakukan, antara lain Jawatex[12], Hanacaraka[13], dan Pandawa[14]. Dari hasil penelitian tersebut ditunjukkan bahwa metode aplikasi Pandawa dengan metode *decision tree* memiliki tingkat akurasi yang tertinggi dibanding aplikasi lainnya. Namun demikian, waktu komputasi yang dibutuhkan oleh metode dalam aplikasi ini juga paling tinggi dibandingkan aplikasi yang lain. Hal ini berbanding lurus dengan banyaknya teks yang ditransliterasikan. Semakin banyak teks yang ditransliterasikan maka akan semakin lama pula waktu pemrosesan yang diperlukan.

Oleh karena itu, penelitian ini bertujuan untuk mempersingkat waktu pemrosesan pada aplikasi Pandawa tanpa perubahan pada algoritma *decision tree*. Dalam penelitian ini diajukan metode komputasi paralel untuk solusi permasalahan tersebut. Komputasi paralel diimplementasikan dengan *multithread* pada pemrograman java untuk melakukan proses transliterasi huruf latin ke aksara jawa secara simultan.

II. METODE PENELITIAN

Desain sistem dari metode untuk mempersingkat waktu komputasi dengan komputasi paralel ini terdiri dari tiga bagian ialah: preprocessing, pemrosesan, dan penggabungan. Bagian *preprocessing* terdiri: perhitungan jumlah kata dalam teks yang akan diproses, pembagian teks masukan menjadi beberapa bagian (halaman), dan pembuatan *thread* sebanyak bagian (halaman) teks. *Preprocessing* ini bertujuan untuk membagi teks huruf latin menjadi beberapa bagian agar dapat dikerjakan secara paralel. Berikutnya adalah bagian pemrosesan yaitu proses transliterasi yang dikerjakan secara paralel dengan *thread* yang telah dibuat. Terakhir adalah bagian penggabungan, dimana pada bagian ini hasil transliterasi yang berupa aksara jawa akan digabungkan kembali dan ditampilan ke pengguna. Diagram desain sistem seperti ditunjukkan pada gambar 1.



Gambar 1. Desain Sistem

A. Penghitungan Jumlah Kata

Penghitungan kata dilakukan dengan memecah kata berdasarkan karakter spasi. Kemudian disimpan dalam *array*. Selanjutnya jumlah kata didapatkan dari panjang *array*.

B. Penghitungan dan Pembagian Halaman

Proses penghitungan jumlah halaman atau pembagi dilakukan untuk menentukan jumlah *thread* yang akan dibuat sehingga banyaknya halaman sama dengan banyaknya *thread* yang dibuat. Teks dibagi setiap 380 kata (diasumsikan sebagai jumlah kata dalam satu halaman). Pembagian dilakukan berdasarkan titik karena dalam aturan penulisan aksara Jawa terdapat aturan pasangan yang mana pasangan memiliki fungsi untuk menghubungkan suku kata yang berakhiran konsonan dengan suku kata berikutnya. Oleh karena itu tidak bisa sembarangan memisahkan kata-perkatanya karena akan menyebabkan terjadinya kekeliruan pada hasil yang diperoleh dan akan menyebabkan penurunan pada tingkat akurasinya. Jika karakter terakhir pada kata terakhir di sebuah halaman bukan titik, maka akan dicari kata selanjutnya yang memiliki karakter terakhir titik.

Proses pencarian karakter titik pada kata selanjutnya ini akan menghasilkan batas baru yang menjadikan satu halaman tidak persis berjumlah 380 kata. Dan proses ini juga menyebabkan kemungkinan bahwa jumlah halaman tidak sama dengan jumlah array halaman yang telah dibuat sebelumnya.

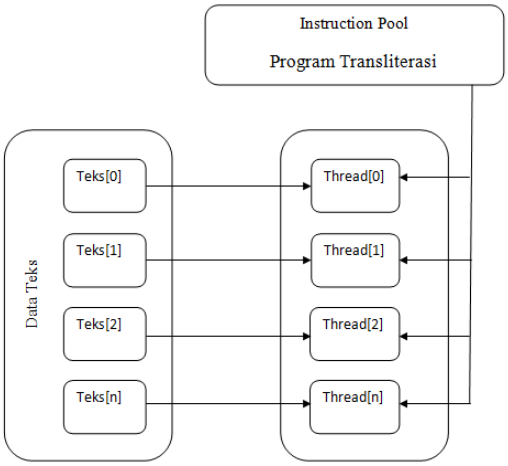
C. Pembuatan Thread

Pembuatan *thread* dilakukan setelah mendapatkan jumlah halaman. Setelah jumlah halaman didapatkan maka dilakukan perulangan berdasarkan indeks halaman dari halaman pertama sampai halaman terakhir. Setelah thread dibuat maka *thread* akan dihidupkan (*start*).

ISSN 2527-1682 (Print)
ISSN 2527-9165 (Online)

D. Proses Transliterasi Menggunakan *Multithread*

Proses selanjutnya adalah proses transliterasi menggunakan metode *Desicion Tree* secara paralel. Pemrosesan transliterasi akan dikerjakan secara paralel menggunakan *thread* yang jumlahnya dibuat sebanyak halaman dokumen seperti ditunjukkan pada gambar 2.



Gambar 2. Alur Kerja Komputasi Paralel Menggunakan Multithread

Beberapa *thread* yang jumlahnya sesuai dengan jumlah halaman. Komputasi paralel yang digunakan adalah model komputasi *single instruction, multiple data stream* (SIMD) yang mana adalah salah satu klasifikasi komputasi paralel klasifikasi Flynn[15].

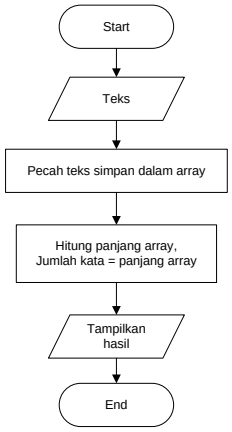
III. HASIL DAN PEMBAHASAN

Pada bagian ini akan akan dibahas proses-proses yang terdapat dalam desain secara logis algoritmik. Kemudian dipaparkan data-data hasil pengujian untuk menguji performa metode yang diajukan.

A. Penghitungan Jumlah Kata

Proses penghitungan kata dijalankan setelah diinputkan teks dengan huruf latin. *Flowchar*t dari proses ini dapat dilihat pada gambar 3. Berikut adalah algoritma proses penghitungan jumlah kata.

1. Teks masukan
   Teks dimasukkan ke dalam teks area. Kemudian program akan melakukan pemecahan teks yang dimasukkan setelah pengguna menekan tombol transliterasi.
2. Pemecahan Teks
   Pemecahan teks dilakukan berdasarkan spasi menggunakan fungsi split dan disimpan ke dalam sebuah array.



Gambar 3. Flowchart Penghitungan Jumlah Kata

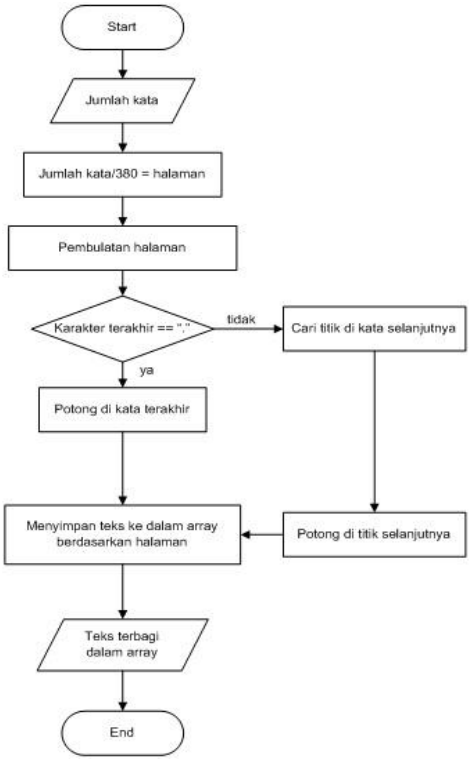
3. Penghitungan Jumlah Kata
   Jumlah kata didapatkan dari panjang *array* dengan menggunakan fungsi `array.length`.

B. Penghitungan dan Pembagian Halaman

Flowchart dari proses penghitungan dan pembagian halaman ini dapat dilihat pada gambar 4. Berikut adalah penjelasan dari diagram alur pada gambar 4:

1. Pembagian teks dengan 380
   Jumlah kata yang telah didapatkan dari proses pemecahan kata (splitting) akan dibagi dengan 380, yang diasumsikan sebagai rata-rata jumlah kata dalam satu halaman dokumen.
2. Pembulatan
   Jika pembagian tidak habis dibagi 380 maka akan dibulatkan keatas. Pembulatan ini menggunakan fungsi `divide()` yang telah disediakan oleh Java dengan mode pembulatan (*rounding mode*) `CEILING` atau keatas.
3. Pembagian teks berdasarkan titik
   Pembagian teks menjadi halaman dilakukan setelah tanda baca titik, karena penulisan aksara jawa berdasarkan suku kata. Jika suku kata berakhiran huruf konsonan dan bertemu huruf konsonan di awal suku kata berikutnya maka digunakan aksara pasangan. Maka untuk pemotongan teks harus dipastikan di akhir kalimat yang ditandai dengan karakter titik setelah kata terakhir pada kalimat tersebut.
4. Penyimpanan teks ke dalam *array* dan penghapusan *array* kosong
   Hasil pembagian teks disimpan dalam *array* dengan urutan *index* sesuai dengan proses pemotongan teks. Panjang *array* (`array.length`) yang digunakan untuk penyimpanan hasil pembagian teks menunjukkan jumlah halaman dokumen yang akan ditransliterasikan. Setelah proses pembagian halaman selesai dan setiap halaman telah berakhiran titik, array yang digunakan sebagai penyimpan teks ini digunakan sebagai acuan untuk penghapusan array kosong. Selanjutnya

dijalankan proses transliterasi secara paralel menggunakan *multithread* dengan jumlah thread sesuai dengan panjang *array* (`array.length`) yang memuat teks.

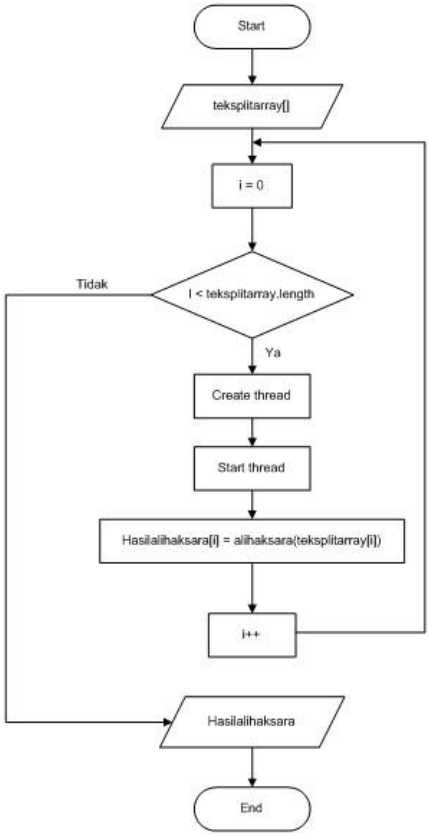


Gambar 4. Flowchart Penghitungan dan Pembagian Halaman

C. Pembuatan *thread* dan proses transliterasi

Pemrosesan transliterasi dilakukan secara paralel dengan *multithread*. Jumlah thread yang dibuat didasarkan pada panjang array (`array.length`) yang digunakan untuk penyimpanan teks. Adapun algoritma dalam proses ini seperti ditunjukkan *flowchart* pada gambar 5. Dari gambar tersebut dapat dilihat bahwa proses transliterasi dengan multithread dengan algoritma sebagai berikut:

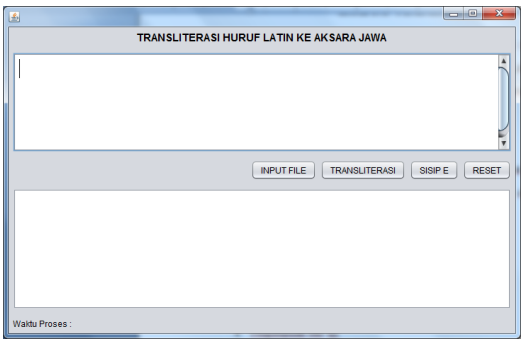
1. Penentuan banyaknya perulangan dalam pembuatan thread berdasarkan panjang *array* (`array.length`) yang memuat teks.
2. Thread yang telah dibuat kemudian dijalankan (*start*) dan digunakan untuk mentransliterasikan teks.
3. Setelah transliterasi selesai maka teks hasil dari beberapa *thread* akan digabungkan kembali dan ditampilkan sebagai hasil dari proses transliterasi.



Gambar 5. Flowchart Proses Transliterasi Menggunakan Multithread

D. Pengujian

Dalam pengujian digunakan data teks latin berbahasa jawa yang diambil dari website www.sastra.org, yang dikelompokkan menjadi 10 halaman, 20 halaman, 30 halaman dan seterusnya sampai 100 halaman dimana setiap kelompok tersebut dijadikan satu dokumen (*file*).



Gambar 6. Tampilan Aplikasi

Adapun tahap-tahap pengujian sebagai berikut: mula-mula aplikasi dijalankan. Kemudian ketika aplikasi telah berjalan seperti ditunjukkan pada gambar 6, teks uji dalam bentuk dokumen (*file*) yang telah disiapkan dimasukkan ke dalam *text area input* dengan klik tombol *input file* kemudian dicari nama dan di folder mana file yang akan digunakan berada. Selanjutnya setelah tombol *transliterasi* ditekan maka proses transliterasi akan dimulai dan hasil alih aksara akan dimunculkan di *text area output*. Pengujian dilakukan dengan cara yang sama terhadap teks 10 halaman, 20 halaman, sampai dengan 100 halaman.

ISSN 2527-1682 (Print)
ISSN 2527-9165 (Online)

Pengujian ini diftekankan pada perbandingan waktu yang diperlukan pada dua proses komputasi yang berbeda, yaitu menggunakan *multithread* dan tanpa *multithread* (sekuensial) dengan pencatatan waktu yang digunakan. Waktu komputasi yang digunakan dicatat kemudian ditabelkan sehingga diperoleh perbandingan catatan waktunya.

E. Hasil

Pengujian pertama digunakan komputer dengan spesifikasi sebagai berikut:

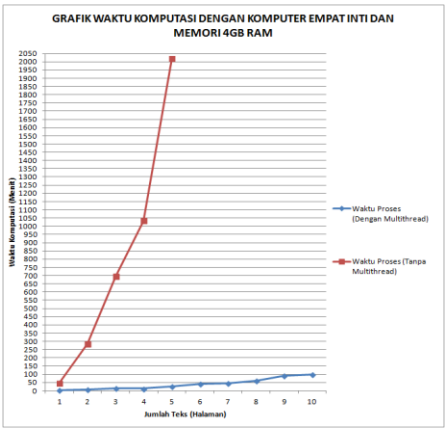
Sistem Operasi : Windows 7 Home Premium 32-bit
Processor : Intel(R) Core(TM) i5-2410M CPU @2.30GHz (4CPUs)
Memory : 4096MB

Adapun hasil pengujian pertama ini seperti ditunjukkan pada tabel 1

Tabel 1. Hasil Pengujian pada Komputer dengan Prosesor Empat Inti dan Memory 4 GB

No	Banyak Halaman	Waktu Komputasi (detik)		Selisih Waktu (detik)
		Dengan Multithread	Tanpa Multithread	
1	10	239,4	2690,0	2450,6
2	20	482,8	17036,0	16553,2
3	30	859,5	41722,0	40862,5
4	40	780,9	61913,0	61132,1
5	50	1651,0	121080,0	119429,0
6	60	2583,0	*Out of Memory*	-
7	70	2704,0	*Out of Memory*	-
8	80	3666,0	*Out of Memory*	-
9	90	5407,0	*Out of Memory*	-
10	100	5849,0	*Out of Memory*	-

Dari tabel 1 dapat dilihat proses tanpa multithread pada teks 60 halaman sampai 100 halaman mengalami kekurangan memori sehingga proses tidak selesai, atau terhenti saat proses berlangsung. *Error* tersebut terjadi karena tidak cukupnya memori untuk mengalokasikan objek dalam java *heap*. Ketika sebuah program java dimulai, *java virtual machine* (JVM) mendapatkan sejumlah memori dari sistem operasi. Dari tabel 1 dihasilkan grafik seperti pada gambar 7.



Gambar 7. Grafik Waktu Komputasi dengan Prosesor Empat Inti dan Memori 4 GB RAM

Pengujian selanjutnya dilakukan pada komputer dengan spesifikasi sebagai berikut:

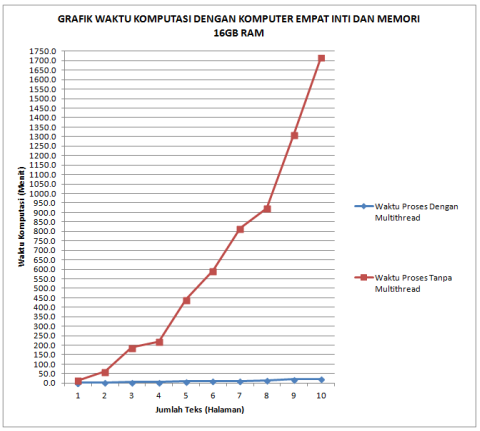
Sistem Operasi : Windows 10 Pro 64-bit
Processor : Intel(R) Core(TM) i5-4460 CPU @3.20GHz (4CPUs)
Memory : 16384MB

Data hasil pengujian kedua yang disajikan pada tabel 2 menunjukkan bahwa dengan dokumen 100 halaman maka selisih waktu pemrosesan sekuensal dengan multithread mencapai 28, 3 jam.

Tabel 2 Hasil Pengujian pada Komputer dengan Prosesor Empat Inti dan Memory 16 GB

No	Banyak Halaman	Waktu Komputasi (detik)		Selisih Waktu (detik)
		Dengan Multithread	Tanpa Multithread	
1	10	81,2	895,3	814,1
2	20	186,5	3715,0	3528,5
3	30	314,6	11242,0	10927,4
4	40	336,4	13146,0	12809,6
5	50	539,5	26501,0	25961,5
6	60	651,6	35618,0	34966,4
7	70	727,6	49033,0	48305,4
8	80	936,5	55396,0	54459,5
9	90	1211,0	78647,0	77436,0
10	100	1289,0	103060,0	101771,0

Tampilan grafik dari data hasil pengujian tabel 2 disajikan pada gambar 8



Gambar 8. Grafik Waktu Komputasi dengan Prosesor 4 Inti dan Memori 16 GB

Pengujian ketiga dilakukan pada komputer dengan spesifikasi sebagai berikut

Sistem Operasi : Windows 7 Professional 64-bit
Processor : Intel(R) Core(TM) i7-6700 CPU @3.40GHz (8 CPUs)
Memory : 8192MB RAM

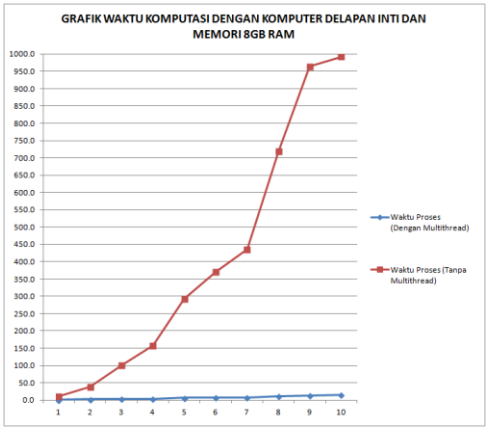
Data hasil pengujian ketiga disajikan pada tabel 3. Dari data tersebut ditunjukkan bahwa dengan prosesor inti 8 dan memori 8 GB selisih waktu komputasi terbesar antara pemrosesan dengan multithread dan tanpa multithread yang terjadi pada dokumen 100 halaman menunjukkan 58658,8 detik. Dimana hasil ini hampir setengahnya dari selisih waktu komputasi yang terjadi

pada prosesor 4 inti dan memori 16 GB dengan jumlah halaman yang sama yaitu 101771,0 detik. Namun demikian waktu 58658,8 detik ini masih cukup besar karena setara dengan 16 jam.

Tabel 3. Hasil Pengujian pada Komputer dengan Prosesor Delapan Inti dan Memory 8GB

No	Banyak Halaman	Waktu Komputasi (detik)		Selisih Waktu (detik)
		Dengan Multithread	Tanpa Multithread	
1	10	58,1	634,5	576,4
2	20	141,6	2340,0	2198,4
3	30	215,6	6043,0	5827,4
4	40	244,4	9474,0	9229,6
5	50	394,5	17607,0	17212,5
6	60	432,2	22263,0	21830,8
7	70	487,0	26130,0	25643,0
8	80	676,2	43119,0	42442,8
9	90	818,5	57843,0	57024,5
10	100	889,2	59548,0	58658,8

Dari tabel 3 diperoleh grafik seperti yang disajikan pada gambar 9.



Gambar 9. Grafik Waktu Komputasi dengan Prosesor 8 Inti dan Memori 8 GB

Pada grafik yang ditunjukkan oleh Gambar 7, 8, dan 9 terdapat perbedaan garis merah dan biru yang sangat jauh. Garis merah (yang menunjukkan waktu proses tanpa *multithread*) pada grafik menunjukkan kenaikan waktu yang signifikan dibanding garis biru (yang menunjukkan waktu proses menggunakan *multithread*). Semakin tinggi garis semakin banyak pula waktu yang dibutuhkan oleh CPU untuk melakukan eksekusi program. Selain itu perbedaan juga dipengaruhi jumlah memori dan jenis CPU yang digunakan. Semakin besar memori maka semakin kecil kemungkinan terjadi *Out of memory* pada komputer. Hal ini dikarenakan memori yang dibutuhkan untuk proses semakin besar.

Untuk menunjukkan seberapa efisien implementasi komputasi paralel *multithread* maka dihitung persentase dari selisih waktu dari hasil percobaan yang telah dilakukan. Penghitungan presentase dilakukan dengan membandingkan nilai dari selisih waktu komputasi dengan nilai dari waktu komputasi sebelum implementasi *multithread* yang mana dapat dilihat pada rumus (1)[16]:

$$\text{Persentase Selisih Waktu} = \left|\frac{N2-N1}{N1} \times 100\%\right| \quad (1)$$

Dimana,

- N1 Nilai Waktu Komputasi tanpa multithread
- N2 Nilai Waktu Komputasi dengan *multithread*

Tabel 4.4 Rata-rata Selisih Waktu Komputasi Perjumlah Halaman

Jumlah Halaman	Persentase Selisih Waktu per Jumlah Halaman			
	core i5 4GB	core i5 16GB	core i7 8GB	Rata-rata
10	91,10%	90,93%	90,84%	90,95%
20	97,17%	94,98%	93,95%	95,36%
30	97,94%	97,20%	96,43%	97,19%
40	98,74%	97,44%	97,42%	97,87%
50	98,64%	97,96%	97,76%	98,12%
60	-	98,17%	98,06%	98,74%
70	-	98,52%	98,17%	98,88%
80	-	98,31%	98,43%	98,91%
90	-	98,46%	98,58%	99,01%
100	-	98,75%	98,50%	99,08%

Dari Tabel 4 didapatkan rata-rata persentase selisih waktu komputasi per jumlah halaman. Dari rata-rata tersebut dapat digunakan untuk mendapatkan rata-rata selisih seluruh waktu komputasi antara komputasi dengan menggunakan multithread dan tanpa multithread. Untuk mendapatkan persentase rata-rata selisih keseluruhan digunakan rumus (2)[17]:

$$Rata - rata\ persentase = \frac{P1+P2+P3+\cdots+Pn}{n} \quad (2)$$

Dimana,

- *P1,* P2, P3 …Pn Persentase selisih waktu pada data ke 1 sampai n
- n jumlah data uji

Dari rumus nilai rata-rata tersebut dapat dihitung nilai rata-rata persentase dari selisih waktu komputasi dan didapatkan bahwa nilai rata-rata persentase selisih waktu komputasi dengan menggunakan *multithread* dan tanpa *multithread* adalah 97,05%.

IV. PENUTUP

Berdasarkan pengujian yang telah dilakukan, maka didapatkan kesimpulan sebagai berikut:

1. Komputasi paralel model pemrograman *multithread* dapat diterapkan untuk mempercepat waktu komputasi pada aplikasi transliterasi huruf latin ke aksara jawa.
2. Implementasi komputasi paralel pada aplikasi transliterasi huruf latin ke aksara jawa ini dapat menghemat waktu proses sebesar 97,05%.

V. REFERENSI

[1] N. J. Smith-Hefner, "Youth Language, Gaul Sociability, and the New Indonesian Middle Class," *J. Linguist. Anthropol.*, vol. 17, no. 2, pp. 184–203, Dec. 2007.

[2] D. E. Subroto, M. D. Rahardjo, and B. Setiawan, "Endangered krama and krama Inggil varieties of the Javanese language," *Linguist. Indones.*, vol. 26, no. 1, pp. 89–96, 2008.
[3] S. Nurdjanah, "Factors that Influence Junior HighSchool Students in Semarang Prefer to Use Bahasa Indonesiathan the Javanese Language," Diponegoro University, 2015.
[4] J. Juliansyah, A. Saragih, and B. Gurning, "Language Shift of the Javanese in Stabat," *Linguist. Terap.*, vol. 12, no. 1, pp. 17–24, 2015.
[5] G. Poedjosoedarmo, "The effect of Bahasa Indonesia as a lingua franca on the Javanese system of speech levels and their functions," *Int. J. Soc. Lang.*, vol. 2006, no. 177, pp. 111–121, 2006.
[6] A. Dharma, "Pembinaan dan Pengembangan Bahasa Daerah," in *Language Maintenance and Shift*, 2011, pp. 8–11.
[7] H. Sulistyorini, Sutijan, and Samidi, "Peningkatan Keterampilan Membaca Dan Menulis Aksara Jawa Melalui Model Pembelajaran Kooperatif Tipe Cooperative Integrated Reading and Composition (Circ)," *Didakt. Dwija Indria*, vol. II, no. 1, pp. 1–6, 2012.
[8] E. A. MARSELIA, "Peningkatan Kemampuan Membaca Huruf Jawa Melalui Strategi Pembelajaran Teams Games Tournament Pada Siswa Kelas V SD N Gawanan Ii Colomadu Tahun Pelajaran 2013 / 2014," Universitas Muhammadiyah Surakarta, 2014.
[9] D. R. Ardiyani, "Keterampilan Membaca Aksara Jawa Melalui Model Quantum Learning Dengan Media Kartu Kata Siswa Kelas Iiia Sdn Petompon 02 Semarang," Universitas Negeri Semarang, Semarang, 2013.
[10] T. O. TENDEAN, "Perancangan Permainan Edukatif Sebagai Sarana Bantu Pembelajaran Aksara Jawa Dengan Studi Kasus Pada Siswa Sekolah Dasar Kelas 3," *Calyptra*, vol. II, no. 2, pp. 1–18, 2013.
[11] A. Indaryatiningsih, "Peningkatan Keterampilan Membaca dan Menulis Aksara Jawa Menggunakan Model Pembelajaran Kooperatif Teknik Make a Match pada Peserta Didik Kelas IV SDN Peleman 01 Sragen Tahun Pelajaran 2012/2013," UNS (Sebelas Maret University), 2013.
[12] E. Utami, J. E. Istiyanto, S. Hartati, M. Marsono, and A. Ashari, "Penerapan Rule Based dalam Membangun Transliterator Jawatex," *Berk. Ilm. MIPA*, vol. 23, no. 1, Jan. 2014.
[13] GigaPurbalingga, "Hanacaraka V1.0 - Software Menulis Huruf Jawa," 2014. .
[14] F. H. Ayumitha, "Transliterasi huruf latin ke dalam aksara Jawa dengan menggunakan decision tree," Universitas Islam Negeri Maulana Malik Ibrahim Malang, 2014.
[15] B. Barney and L. Livermore, "Introduction to Parallel Computing," 2018. .
[16] Microsoft, "Menghitung Persentase," 2018. .
[17] Microsoft, "Menghitung Rata-rata dari Grup Angka," 2018. .